Session d'exercices – Structures et pointeurs
Echantillonage

En traitement du signal, on sait qu'un signal échantilloné a une certaine fréquence d'échantillonage peut être parfaitement reconstruit si la fréquence du signal est strictement inférieure à la moitié de la fréquence d'échantillonage. On appelle cette fréquence maximale la fréquence de Nyquist. Dans cet exercice, nous définirons deux structures représentant les caractéristiques d'un signal sinusoïdal simple et d'un échantillonage et utiliserons ensuite ces structures pour vérifier la théorie en échantillonant des signaux de fréquences diverses. A noter que dans cet exercice, un signal complet est défini comme étant une somme de signaux sinusoïdaux.

Si la théorie sous-jacente n'est plus très claire pour vous, vous trouverez des informations sur les pages suivantes :

1. Signaux sinusoïdaux
2. Echantillonage
3. Théorème d'échantillonage

Si la théorie est claire pour vous, vous pouvez passer aux exercices :

1. [Difficulté : *] Structures de base
   Ecrivez les structure `Sinusoide` et `Echantillonage`
   La première doit contenir 3 champs représentant les 3 constantes inhérentes d'un signal sinusoïdal pur : la fréquence, l'amplitude et le déphasage. La seconde doit contenir 2 champs représentant l'instant auquel l'échantillonage commence et sa fréquence.

2. [Difficulté : **] Sinusoïde pure
   Écrivez la fonction
   `double sin_pure(double t, struct Sinusoide s)`,
   qui prend en argument un réel et une structure Sinusoïde et retourne la valeur de la sinusoïde à un instant t. Si l'utilisation des 3 constantes d'une sinusoïde n'est pas claire, se référer aux pages de théorie plus haut.

3. [Difficulté : **] Signal
   Écrivez la fonction
   `double signal(double t, int n, struct Sinusoide signal[n])`,
   qui prend en argument un réel et un tableau de sinusoïdes de taille `n` et retourne l'évaluation du signal au temps `t`. Il s'agit donc de sommer les évaluations des sinusoïdes à travers le tableau passé en argument en utilisant la fonction précédente.

4. [Difficulté : **] Echantillonnage
   Écrivez la fonction
   `void echantillonne(int n, struct Sinusoide s[n], struct Echantillonage* echant, int nb_echant, double echantillons[nb_echant])`,
   Cette fonction prend en argument un Signal, représenté par un tableau de Sinusoïdes, un Echantillonnage et un tableau de réels où seront stockés les échantillons du signal. Pour ceci, il faut donc évaluer le signal `nb_echant` fois en partant du temps `t0` donné par la structure `echant` et à une fréquence donnée par la même structure.

5. [Difficulté : *] Sinus cardinal
   Écrivez la fonction
   `double sinc(double x)`,
   Qui retourne simplement le sinus cardinal $sinc(x)$ d'un réel passé en argument. Pour rappel, $sinc(x) = sin(\pi * x)/\pi * x$ et $sinc(0) = 1$.
6. [Difficulté : **] Reconstruction du signal
   Écrivez la fonction
   `double reconstruction(double t, struct Echantillonage echant, int nb_echant, double echantillons[nb_echant])`,
   Qui, pour un Echantillonage et un tableau d'échantillons (typiquement calculés au préalable par la fonction `echantillone`), retourne la valeur du signal reconstruit à l'instant t. Ceci est fait n utilisant la formule standard de reconstruction d'un signal, modifiée pour correspondre à cet exercice :
   `valeur[i] = echantillons[i] * sinc(echant.fe * (t-echant.t0) - i );`.
   Il s'agit donc de sommer ces valeurs à travers le tableau d'échantillons passé en argument.

Une fois toutes ces fonctions écrites, il est temps de les tester.

[Difficulté : ***] Préparation du `main()`
Afin de vérifier et expérimenter avec le théorème d'échantillonnage, vous devez :

1. Créer un Signal, somme de sinusoïdales, quelconque. Rapelez-vous qu'un tableau de structures peut être initialisé comme :
   ```
   struct Sinusoide s[] ={
   { 1.0 , 2.0, 0.0 }, //première sinusoïde pure
   { 0.5 , 4.0, 0.1 } //etc.
   };
   ```
2. Créer deux Echantillonnages avec le même temps initial et le même nombre d'échantillons mais avec des fréquences différentes. L'un devrait avoir une fréquence d'échantillonnage supérieure à 2 fois la fréquence du signal de base et l'autre une fréquence inférieure. Pour rappel, la fréquence d'un signal sinusoïdal est la fréquence la plus élevée parmi les sinusoïdes pures qui sont sommées. Dans l'exemple ci-dessus, le fréquence du signal est donc 4.0 et il faudra donc une fréquence d'échantillonnage supérieure à 8.0.
3. Echantilloner le signal de base à l'aide des deux Echantillonnages et de la fonction `echantillonne`. Cette opération vous donnera deux tableaux d'échantillons qui serviront pour la suite.
4. Reconstruire le signal de base à l'aide des deux Echantillonnages et de la fonction `reconstruction`. On vous demande ici de simplement évaluer la reconstruction à un temps quelconque. Rappelez-vous, cette fonction ne fait qu'évaluer le signal reconstruit en un point et ne crée pas de Signal à proprement parler. Pour visualiser le résultat de la reconstruction complète, il faut donc appeler cette fonction à plusieurs instants et comparer le résultat avec le signal original. C'est le sujet de l'exercice suivant.

[Difficulté : ***] Visualisation des résultats
Maintenant que vos structures ont été initialisées, vous êtes prêts à tester vos fonctions. Ceci se fera en deux phases :

1. Commencez par "dessiner" les résultats en texte sous forme de colonnes où chaque ligne contient : le temps t, le signal évalué au temps t et les deux reconstructions au même temps t. Pour ce faire, écrivez une boucle `for` allant de `t0` à un temps quelconque et avançant par

pas de temps (0.1 par exemple).
On s'attend à quelque chose comme :

```
0.000000 0.189358 0.189358 0.189358
0.100000 0.755640 0.755640 0.898548
0.200000 0.257756 0.257756 0.273642
//etc.
```

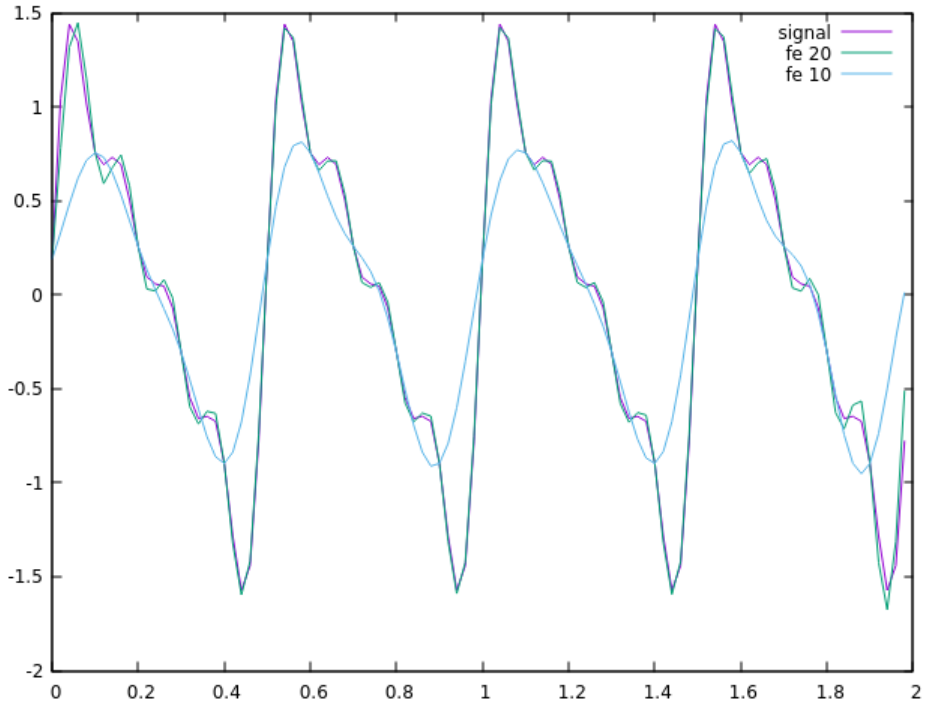
Note : On pourrait également afficher une légende en première ligne, quelque chose comme `t X(t) X_I(t) X_II(t)`,
mais pour la deuxième partie il faut que l'affichage de contienne que des nombres.

2. Vous allez maintenant vraiment dessiner les courbes de ces signaux à l'aide de `gnuplot` directement depuis le terminal. Ce programme, présent sur la grande majorité des distributions Linux, lit des fichiers contenant des valeurs numériques et trace des courbes basées sur ces valeurs qui doivent être écrites en colonne. Vous comprenez maintenant pourquoi l'affichage se faisait en colonne au point précédent. Cependant, notre programme affiche simplement ces valeurs dans le terminal, pas dans un fichier. Heureusement, il existe une façon très simple de résoudre ce problème. Dans le terminal, déplacez-vous jusqu'au répertoire contenant votre programme et, si celui-ci s'appelle `echantillonnage` par exemple, tapez simplement `./echantillonnage > data.txt`. Cette commande éxécute votre programme et *redirige* sa sortie (l'affichage du point précédent) vers le fichier "data.txt". Vérifiez que l'écriture a fonctionné en tapant `cat data.txt` ce qui devrait afficher la même chose que votre programme.

   Ceci fait, il suffit de taper `gnuplot` dans le terminal et dans gnuplot de taper :

```
> set style data line
> plot "data.txt" u 1:2 t "signal", "data.txt" u 1:3 t "fe 20",
"data.txt" u 1:4 t "fe 10"
```

   Où vous remplacerez "20" et "10" par les fréquences d'échantillonnages que vous aurez utilisé. Avec un affichage sur 100 points, vous devriez obtenir un graphique similaire à celui-ci :



Comme vous pouvez le voir, le signal reconstruit à base d'un échantillonage d'une fréquence deux fois supérieure à celle du signal original suit presque parfaitement le signal de base tandis que le second échantillonnage est assez différent. Ceci correspond bien à la théorie et prouve que notre modèle ainsi que notre programme est correct.
Note : On remarque que le signal reconstruit ne suit pas exactement la signal de base. Ceci est du au fait que le signal est reconstruit à partir d'une somme finie de sinus cardinaux.
Rapide explication de la commande gnuplot :

— Par défaut, gnuplot ne trace que des points. La commande `set style data line` indique à gnuplot que vous désirez tracer des courbes à la place.
— `u 1:2`,`u 1:3` et `u 1:4` indiquent à gnuplot quelles colonnes utiliser. u 1 :3, par exemple indique qu'il faut utiliser la colonne 1 comme abscisse et la colonne 3 comme ordonnée.
— `t "signal"`, `t "fe 10"` et `t "fe 20"` indiquent simplement la légende à afficher pour chacune des courbes.

Pour plus d'information sur la commande `plot`, vous pouvez consulter la documentation officielle ou taper `man gnuplot` dans un terminal.